Cheryl Melwani and Devan Venkataraman

COMP40 - HW 7

April 22, 2021

HW 7 - Profiling

Testing:

Benchmark	*Time*	*Instructions*	*Rel to start*	*Rel to prev*	*Improvement*
Small - midmark Large - sandmark	6.14 s 156.46 s	2.958×10^{10}	1.000 1.000	1.000 1.000	None, original code
Small - midmark Large - sandmark	5.08 s 130.54 s	2.775×10^{10}	0.8274 0.8343	0.8274 0.8343	Compile using -01
Small - midmark Large - sandmark	4.98 s 127.64 s	2.761×10^{10}	0.8111 0.8158	0.9803 0.9778	Compile using -02
Small - midmark Large - sandmark	3.39 s 88.67 s	2.469×10^{10}	0.5521 0.5667	0.6807 0.6947	Changed the structure of mapped segments from a Hanson sequence to a local segment struct that stores uint32_t's
Small - midmark Large - sandmark	3.36 s 86.92 s	2.462×10^{10}	0.5472 0.5555	0.9912 0.9803	Changed the structure that's holding the unmapped ID's from a Hanson sequence to using our local segment struct that stores uint32_t's
Small - midmark Large - sandmark	2.52 s 65.00 s	2.230×10^{10}	0.4104 0.4154	0.7500 0.7478	Replaced outer Hanson sequence of segments with local sequence struct that holds an array of our segment struct.
Small - midmark Large - sandmark	2.54 s 64.83 s	2.229×10^{10}	0.4137 0.4143	1.0080 0.9974	Compiled all of the code in different files into the main um.c file. The time that the

					compiler takes to link all of the files gets reduced.
Small - midmark Large - sandmark	2.48 s 62.92 s	2.228×10^{10}	0.4039 0.4021	0.9764 0.9705	Removed calls to bitpack_new when reading in instructions by using getc and shifting instead
Small - midmark Large - sandmark	1.16 s 29.92 s	1.014×10^{10}	0.1889 0.1912	0.4677 0.4755	Kcachegrind showed that 50% of program time was spent in bitpack_get. So, we removed all calls to bitpack_get by using shifting instead.
Small - midmark Large - sandmark	0.738 s 19.266 s	5.5886×10^{9}	0.1202 0.1231	0.6362 0.7157	Kcachegrind also showed we spent significant time in segmentedLoad, which was used to read in instructions. We were able to avoid the call by making iterative array index calls instead.
Small - midmark Large - sandmark	0.489 s 13.537 s	3.5642×10^{9}	0.0796 0.0865	0.6626 0.7026	Kcachegrind showed that segmentedLoad and Store required lots overhead due to the calls of Seq_get and Seq_put. We removed those function calls and used array indexing instead.
Small - midmark Large - sandmark	0.381 s 10.839 s	2.8901×10^{9}	0.0621 0.0693	0.7791 0.8007	Removed all unnecessary calls to instruction functions (from executeHelper) and instead performed the instruction in the switch statement.

At this point, any changes we tried such as removing assert statements and reducing function calls had no measurable impact on performance on either of our benchmarks.